Docket: 43876-156 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
HANSEN et al.	:	Confirmation Number: 5955
Application No.: 10/757,866	:	Group Art Unit: 2183
Filed: January 16, 2004	:	Examiner: Eric Coleman

For: METHOD AND SOFTWARE FOR STORE MULTIPLEX OPERATION

DECLARATION OF KORBIN VAN DYKE

I, Korbin Van Dyke, state that:

Personal Background

1. I am an electrical/computer engineer with a BS and an MS in Electrical Engineering and Computer Science, and more than 20 years of industry and academic experience in computer architecture, processor microarchitecture, and logic design of complex VLSI systems, particularly microprocessors. I am a co-inventor on more than 50 issued U.S. patents, relating to a variety of subjects, including multiple ISA execution, hardware optimization for binary translation, video decoding ISA optimization, compatible ISA implementation techniques, speculative instruction execution, branch prediction, compatible segmentation and paging, standard PC sub-system virtualization, address generation, and logarithmic calculation.

2. I received a Master's of Science degree in Electrical Engineering and Computer Science (MS EECS) from the University of California (UC) Berkeley in 1982. Prior to receiving my Master's of Science degree, I received a Bachelor's of Science degree in Electrical Engineering and Computer Science from UC Berkeley in 1980.

3. While working toward my MS EECS degree, I also worked at UC Berkeley as a Research Assistant on the logic specification, logic verification, functional simulation, circuit design, and layout verification of an implementation of a Reduced Instruction Set Computer (RISC-I).

4. After graduating from UC Berkeley, I worked at various engineering and management positions from 1982 until 2001 that included involvement in the specification, implementation, and debugging of complex computer architecture systems. Further selected details of my work during this period are set forth below in paragraphs 5 through 8.

5. From June 1982 to October 1987, I worked at VLSI Technology in San Jose, California as a VLSI Systems Engineer. I specified and designed a memory interface for a multi-mode display interface chip. I defined the instruction set architecture and microarchitecture of a programmable digital signal processor (DSP), and led a team that built and evaluated a microprocessor implementation of the DSP.

6. From October 1987 to May 1996, I worked at Nexgen Microsystems and remained with the company after it was acquired by Advanced Micro Devices (AMD) in January 1996, in various offices in San Jose and Milpitas, California. I designed the microarchitecture and logic details of the pipeline control for a superscalar, out-of-order, x86-compatible, multi-chip microprocessor, the Nx586. I also micro-architected and logic designed the branch prediction and multiple stream instruction fetch hardware of the Nx586. I oversaw various aspects of the physical design of the Nx586 (and its predecessors), including synthesis from Verilog code, place and route, timing closure, and layout versus schematic/netlist verification. I led a team that verified architectural correctness of the Nx586, and I led a team that debugged the Nx586 multi-chip silicon. I investigated several micro-architectural tradeoffs with respect to future high-performance x86 processors.

7. From May 1996 to May 2000, I worked at Chromatic Research and remained with the company after it was acquired by ATI Research Silicon Valley in November 1998, in Sunnyvale, and then Santa Clara, California. I oversaw the development and made personal contributions to the instruction set architecture of a dual-instruction-set processor (x86-compatible and RISC), including specifications for efficient transitions between the instruction sets, architectural hooks for efficient dynamic binary translation from the x86 instruction set to the RISC instruction set, and SIMD arithmetic and special hardware assists for media processing operations. I led a team that developed the pipeline control section of the processor (including Verilog coding and synthesis), and oversaw a team that developed the physical design of portions of the processor (place and route).

8. From June 2000 to December 2001, I worked as the Director of VLSI Development at XStream Logic and remained with the company as an Architect (after the company changed its name to Clearwater Networks in October 2001) in Los Gatos, California. As a VLSI Director, I led a team that was responsible for implementing a complex network processor from a set of specifications to silicon, using a simulatable high-level language description of the processor as a starting point. The team used a tool set that enabled automatic conversion of the high-level language into Verilog for simulation and synthesis, and then placement and routing. As an Architect, I defined and documented various architectural approaches for future network processors.

9. I am currently a sub-contractor of PatentVentures, an intellectual property consulting company with offices in San Jose, California and Austin, Texas. I have been a sub-contractor for or employed by PatentVentures as an intellectual property consultant since January

2002 and became a registered U.S. Patent Agent (Reg. No. 52,313) in August 2002. I am also owner/operator of Van Dyke Consulting (formerly known as Korbin S Van Dyke Consulting), since May 2004, an intellectual property and technical services consulting company with an office in Sunol, California.

10. A copy of my CV is attached as Exhibit A.

Summary of My Opinions

11. In preparation of this declaration I have reviewed U.S. Patent Application Serial No. 10/757,866. I have also reviewed U.S. Patent Nos. 6,295,599 and 5,742,840 (respectively the '599 and '840 patents) that the 10/757,866 patent application indirectly claims priority to, as well as appendices to the '599 and '840 patents (the Zeus and Terpsichore System Architecture manuals, respectively, and hereinafter referred to respectively as the Zeus and the Terpsichore manuals). I have reviewed the Office Action for the 10/757,866 patent application mailed on February 21, 2007, including the paragraphs on pages 8-9 that discuss the Response to Arguments and particularly the Examiner's conclusion that the priority for the claimed invention does not extend to the '599 or the '840 patents, since features of the claimed invention are not taught or supported by the '840 or '599 patents. My understanding is that the features of the claimed invention are taught and supported by complying with the written description requirement and the enablement requirement. My understanding of the written description requirement is that a patent disclosure must describe the claimed invention in sufficient detail that one of ordinary skill in the art can reasonably conclude that the inventor had possession of the claimed invention at the time of filing the patent disclosure. My understanding of the enablement requirement is that the patent disclosure must contain sufficient information regarding the subject matter of the claims to enable one of ordinary skill in the pertinent art to make and use the claimed invention. I further understand that whether the enablement requirement is met depends on whether undue experimentation is necessary for one of skill in the art to practice the invention in light of the patent disclosure.

12. Based on my review of the materials identified in paragraph 11, it is my opinion that:

> (i) the disclosure of the '599 patent and the '840 patent each indicate that the inventors were in possession of the claimed (as amended) "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value and to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements of the 10/757,866 patent application as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent; and

(ii) the disclosures of the '599 patent and the '840 patent each would have enabled a person of ordinary skill in the art to make and use, without undue experimentation, the claimed (as amended) "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements of the 10/757,866 patent application as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent.

13. The disclosure of the '840 patent provides detailed information and description that I believe indicates that the inventors were in possession of the claimed (as amended) "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements of the 10/757,866 patent application, and that I further believe would have enabled a person of ordinary skill in the art to make and use the claimed invention without undue experimentation. On at least pages 150-157 of the Terpsichore manual (describing Store and Store Immediate instructions, including Store Multiplex and Store Multiplex Immediate forms) there are detailed descriptions of the aforementioned claim elements.

14. The disclosure of the '599 patent provides detailed information and description that I believe indicates that the inventors were in possession of the claimed (as amended) "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements of the 10/757,866 patent application, and that I further believe would have enabled a person of ordinary skill in the art to make and use the claimed invention without undue experimentation. On at least pages 123-125 and 128-130 of the Zeus manual (describing Store and Store Immediate instructions, including Store Multiplex and Store Multiplex Immediate forms) there are detailed descriptions relating to the aforementioned claim elements.

15. A detailed explanation of the basis for my opinions is set forth in the remainder of this declaration.

Detailed Basis for My Opinions

Level of Ordinary Skill in the Art:

16. Based on my review of the 10/757,866 patent application (referred to as the "media processor patent application"), I believe that the media processor patent application pertains to the technology of microprocessor design and microprocessor systems.

17. In my opinion a person of ordinary skill in the art of microprocessor design and microprocessor systems would possess a bachelor's degree in electrical engineering or computer science and have approximately five years of experience in the field of microprocessor design and microprocessor systems. Alternatively, an equivalent person could have had a master's degree in electrical engineering or computer science and have approximately two or more years of experience in the field of microprocessor design and microprocessor systems. This person would readily understand the conceptual design of a microprocessor and a microprocessor execution unit. The person of ordinary skill would have had access to a library of technical publications, periodicals, and textbooks.

Analysis of the disclosure of the '599 and '840 patent disclosure:

The disclosures of the '599 patent and the '840 patent each describe the claimed "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" of the 10/757,866 patent application in sufficient detail that a person of ordinary skill in the art could reasonably conclude the inventors were in possession of the claimed invention, and that a person of ordinary skill in the art would have been enabled to make and use the claimed invention without undue experimentation as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent:

18. Claim 1 (as amended) of the 10/757,866 patent application recites various elements:

(a) decoding a single instruction for writing data to memory

(b) based on a mask and data contained in at least one register

(c) the mask comprising fields that each correspond to a field of the data contained in the at least one register;

(d) detecting some of the fields of the mask as having a predetermined value

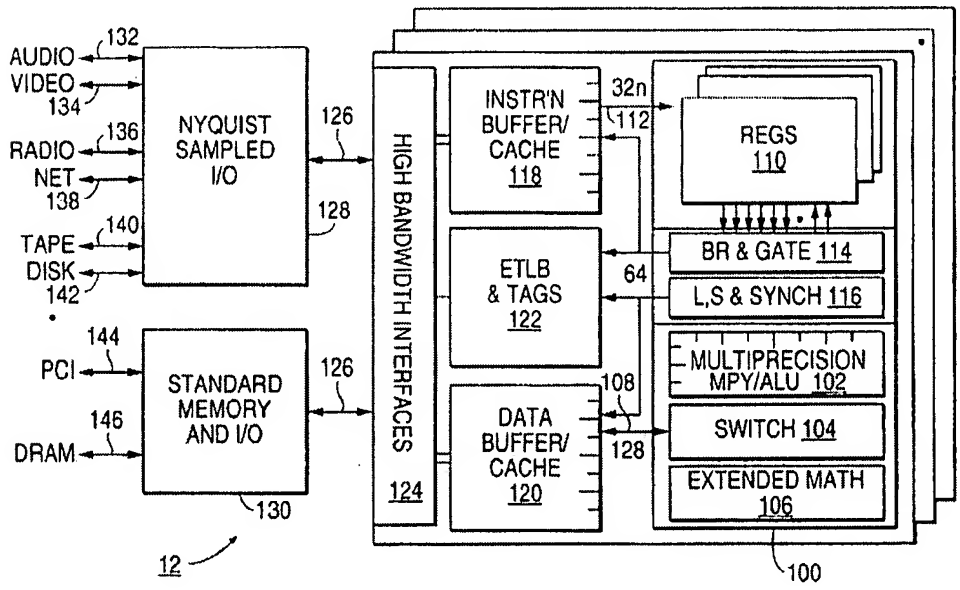
(e) to identify corresponding fields of the data contained in the at least one register as write-enabled data fields; and

(f) writing the write-enabled data fields to a specified memory location

19. For brevity, the following analysis focuses on and provides details relating to the '840 patent, while reciting summary information pointing out where similar descriptive information is provided in the '599 patent.

20. The '840 patent describes structure of a general purpose, programmable media processor. For example, Fig. 7 (reproduced below), illustrates an execution unit 100 having an ALU 102 that performs all logical and simple arithmetic operations (see '840, column 7, lines 8-11 and column 11, lines 50-60). Fig.7 also illustrates a register file 110, to store and transmit data streams to and from the execution unit 100, that includes 64 general purpose registers (see '840, column 12, lines 56-64). Fig.7 also illustrates a combined instruction buffer/cache 118 to store instructions (see '840, column 16, lines 51-67), and data buffer/cache 120 to store data received to and from the execution unit 100 and register file 110 (see '840, column 17, lines 1-14). Several pipeline organizations of the general purpose media processor are described (see '840 Fig. 11 and column 17, lines 15-52, as well as '840 Fig. 12 and column 17, lines 53-67). The '840 patent recites that an instruction set for the general purpose media processor is described by the Microfiche Appendix (see '840 column 13, lines 21-24). The Microfiche Appendix of the '840 patent is referred to herein as the Terpsichore manual. Similarly, the '599 patent describes structure of a general purpose, programmable processor for broadband applications (see, for example, Fig. 1 and associated descriptive text, such as column 4, line 30 to column 5, line 56). Additionally, the '599 patent includes and refers to a Microfiche Appendix that describes, for example, various pipeline organizations and an instruction set for a general purpose processor for broadband applications. The '599 patent Appendix is referred to herein as the Zeus manual.

FIG. 7



21. The Terpsichore manual describes all of the elements of Claim 1, on at least pages 150-157 (attached as Exhibit B) with additional information on page 24 (attached as Exhibit C). Paragraphs 22-28 of this declaration discuss selected portions of those pages. Paragraphs 29-35 of this declaration describe how the elements of Claim 1 (as amended) are described at least by Terpsichore manual pages 24 and 150-157. The Zeus manual describes all of the elements of Claim 1, on at least pages 19-20, 123-125, and 128-130.

22. The Terpsichore manual describes several variations of Store Immediate instructions, including several Store Multiplex Immediate forms: "S.MUX.64.B.A.I" and "S.MUX.64.L.A.I", as described on page 154, and reproduced below (annotations added):

Store Immediate

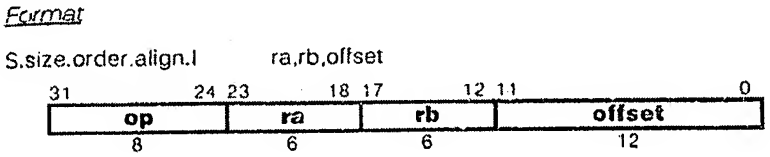
These operations add the contents of a register to a sign-extended immediate value to produce a virtual address, and store the contents of a registerinto memory.

Operation codes

S.8.I[47]	Store byte immediate
S.16.B.A.I	Store double big-endian aligned immediate
S.16.B.I	Store double big-endian immediate
S.16.L.A.I	Store double little-endian aligned immediate
S.16.L.I	Store double little-endian immediate
S.32.B.A.I	Store quadlet big-endian aligned immediate
S.32.B.I	Store quadlet big-endian immediate
S.32.L.A.I	Store quadlet little-endian aligned immediate
S.32.L.I	Store quadlet little-endian immediate
S.64.B.A.I	Store octlet big-endian aligned immediate
S.64.B.I	Store octlet big-endian immediate
S.64.L.A.I	Store octlet little-endian aligned immediate
S.64.L.I	Store octlet little-endian immediate
S.128.B.A.I	Store hexlet big-endian aligned immediate
S.128.B.I	Store hexlet big-endian immediate
S.128.L.A.I	Store hexlet little-endian aligned immediate
S.128.L.I	Store hexlet little-endian immediate
S.AAS.64.B.A.I	Store add-and-swap octlet big-endian aligned immediate
S.AAS.64.L.A.I	Store add-and-swap octlet little-endian aligned immediate
S.CAS.64.B.A.I	Store compare-and-swap octlet big-endian aligned immediate
S.CAS.64.L.A.I	Store compare-and-swap octlet little-endian aligned immediate
S.MAS.64.B.A.I	Store multiplex-and-swap octlet big-endian aligned immediate
S.MAS.64.L.A.I	Store multiplex-and-swap octlet little-endian aligned immediate
S.MUX.64.B.A.I	Store multiplex octlet big-endian aligned immediate
S.MUX.64.L.A.I	Store multiplex octlet little-endian aligned immediate

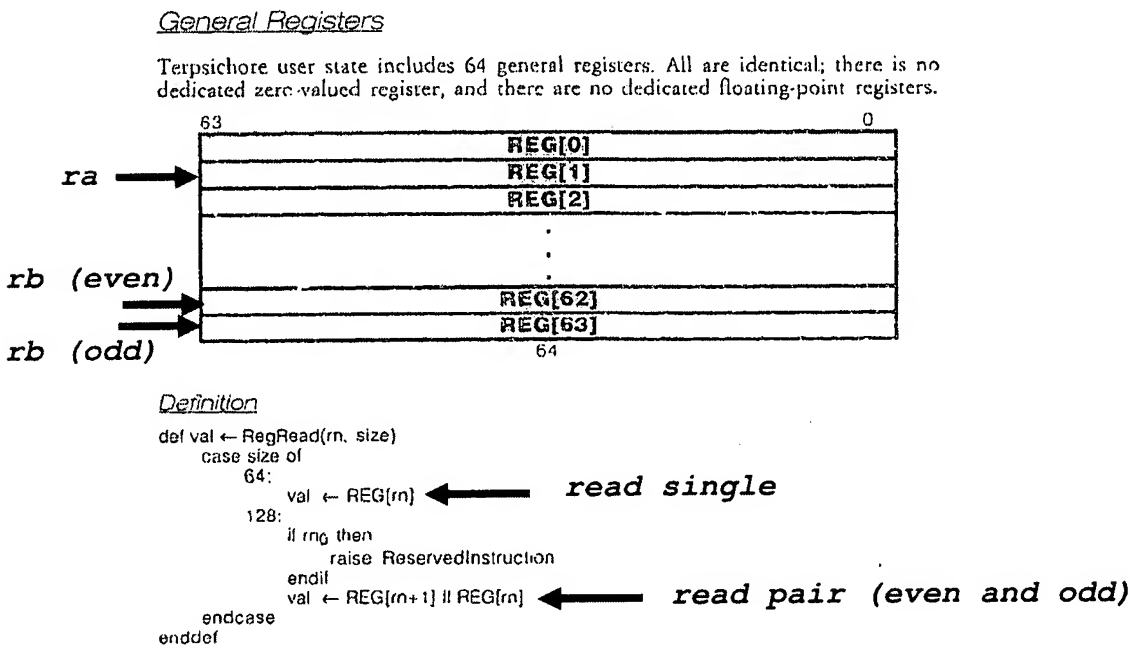
One of ordinary skill in the art would readily understand that instances of instructions, including the foregoing Store Immediate instructions, are, under some circumstances, stored in combined instruction buffer/cache 118 of Fig. 7.

23. The Terpsichore manual, on page 155, describes an instruction format for the Store Immediate instructions, reproduced below:



The operands of the Store Immediate instruction are '`ra`', '`rb`', and '`offset`'. As described in more detail in paragraphs 24-27 of this declaration, a virtual address is formed by adding contents of a register specified by the '`ra`' operand with a sign extension of the '`offset`' operand. Memory at the virtual address is read. Contents of an even-numbered register specified by the '`rb`' operand selectively controls multiplexing, bit-for-bit, between what was read at the virtual address from the memory and contents of an odd-numbered register also specified by the '`rb`' operand. The multiplexing result is then written to the memory at the virtual address.

24. The Terpsichore manual, on page 24, describes the registers referenced as operands to instructions, and also describes associated definition code (hereinafter referred to as pseudo-code) that describes reading a single one of the registers and reading a pair of the registers, as reproduced below (with annotations illustrating examples for reading a single register specified by an '`ra`' operand of '1' and a register pair specified by an '`rb`' operand of '62'):



The foregoing registers are included in register file 110 of Fig. 7 of the '840 patent.

25. The Terpsichore manual, on pages 155-157, describes a definition of Store Immediate instructions, including several forms of Store Multiplex Immediate instructions, with pseudo-code. The pseudo-code begins with descriptions of decoding the Store Immediate instructions, reproduced below (with annotations highlighting description relevant to the Store Multiplex Immediate forms):

Definition

```
def StoreImmediate(op,ra,rb,offset) as
    case op of
        S8I,
        S16LI, S16LAI, S16BI, S16BAI,
        S32LI, S32LAI, S32BI, S32BAI,
        S64LI, S64LAI, S64BI, S64BAI,
        S128LI, S128LAI, S128BI, S128BAI:
            function ← NONE
        SAAS64BAI, SAAS64LAI:
            function ← AAS
        SCAS64BAI, SCAS64LAI:
            function ← CAS
        SMAS64BAI, SMAS64LAI:
            function ← MAS
        SMUX64BAI, SMUX64LAI:
            function ← MUX
    endcase
    case op of
        S8I:
            size ← 8
        S16LI, S16LAI, S16BI, S16BAI:
            size ← 16
        S32LI, S32LAI, S32BI, S32BAI
            size ← 32
        S64LI, S64LAI, S64BI, S64BAI, SAAS64BAI, SAAS64LAI,
        SCAS64BAI, SCAS64LAI, SMAS64BAI, SMAS64LAI, SMUX64BAI, SMUX64LAI:
            size ← 64
        S128LI S128LAI, S128BI, S128BAI:
            size ← 128
    endcase
    case op of
        S8I,
        S16LI, S16LAI, S16BI, S16BAI,
        S32LI, S32LAI, S32BI, S32BAI,
        S64LI, S64LAI, S64BI, S64BAI,
        SAAS64BAI, SAAS64LAI:
            rsize ← 64
        SCAS64BAI, SCAS64LAI, SMAS64BAI, SMAS64LAI, SMUX64BAI, SMUX64LAI:
            rsize ← 128
        S128LI, S128LAI, S128BI, S128BAI:
            rsize ← 128
    endcase
    case op of
        S8I:
            align ← undefined
        S16LI, S32LI, S64LI, S128LI,
        S16BI, S32BI, S64BI, S128BI:
            align ← false
        S16LAI, S32LAI, S64LAI, S128LAI,
        S16BAI, S32BAI, S64BAI, S128BAI,
        SAAS64BAI, SAAS64LAI, SCAS64BAI, SCAS64LAI,
        SMAS64BAI, SMAS64LAI, SMUX64BAI, SMUX64LAI:
            align ← true
    endcase
    case op of
        S8I:
            order ← undefined
        S16LI, S32LI, S64LI, S128LI,
        S16LAI, S32LAI, S64LAI, S128LAI,
        SAAS64LAI, SCAS64LAI, SMAS64LAI, SMUX64LAI:
            order ← L
        S16BI, S32BI, S64BI, S128BI,
        S16BAI, S32BAI, S64BAI, S128BAI,
        SAAS64BAI, SCAS64BAI, SMAS64BAI, SMUX64BAI:
            order ← B
    endcase
```

Thus, decoding of a Store Multiplex Immediate results in '`function`' set to '*MUX*', '`size`' set to '*64*', '`rsize`' set to '*128*', and '`align`' set to '`true`'.

26. The Terpsichore manual pseudo-code for the Store Immediate instructions continues with descriptions of the operation of the instructions based on the decoding, as reproduced below (with annotations highlighting description relevant to the Store Multiplex Immediate forms):

```
[1] ➔ a ← RegRead(ra, 64)
[2] ➔ VirtAddr ← a + (offset11^50 || offset)
[3] ➔ if align then
          if (VirtAddr and ((size/8)-1)) ≠ 0 then
              raise AccessDisallowedByVirtualAddress
          endif
      endif
[4] ➔ m ← RegRead(rb, rsize)
      case function of
          NONE:
              StoreMemory(VirtAddr,size,order,m_size-1..0)
          AAS:
              b ← LoadMemory(VirtAddr,size,order)
              StoreMemory(VirtAddr,size,order,m_63..0+b)
              RegWrite(rb, 64, b)
          CAS:
              b ← LoadMemory(VirtAddr,size,order)
              if (b = m_63..0) then
                  StoreMemory(VirtAddr,size,order,m_127..64)
              endif
              RegWrite(rb, 64, b)
          MAS:
              b ← LoadMemory(VirtAddr,size,order)
              n ← (m_127..64 & m_63..0) | (b & ~m_63..0)
              StoreMemory(VirtAddr,size,order,n)
              RegWrite(rb, 64, b)
[5] ➔     MUX:
              b ← LoadMemory(VirtAddr,size,order)
              n ← (m_127..64 & m_63..0) | (b & ~m_63..0)
              StoreMemory(VirtAddr,size,order,n)
      endcase
  enddef
```

Processing of a decoded Store Multiplex Immediate instruction begins by reading a 64-bit value from a register specified by the '*ra*' operand into '*a*' (see annotation [1]), for example reading REG[1] when '*ra*' is 1. Processing continues by computing a virtual address as '*VirtAddr*' set to the sum of '*a*' and a sign extension of the '*offset*' operand (see annotation [2]), followed by an address alignment check (see annotation [3]). Processing continues by reading a 128-bit value ('*rsize*' is '*128*') from a pair of registers specified by the '*rb*' operand into '*m*' (see annotation [4]). The lower 64 bits of '*m*' contain contents of the even-numbered register specified by '*rb*', and the upper 64 bits of '*m*' contain contents of the odd-numbered register specified by '*rb*'. For example, if '*rb*' is 62, then REG[62] is read into the lower 64 bits of '*m*' and REG[63] is read into the upper 64 bits of '*m*'. Processing continues at the '*MUX*' label (see annotation [5]) ('*function*' is '*MUX*'), by reading a 64-bit value ('*size*' is '*64*') from memory at an address specified by '*VirtAddr*' into '*b*'. A store value is then computed as '*n*' based on '*b*', as well as the lower 64 bits of '*m*' and the upper 64 bits of '*m*'. Processing then continues by writing the store value '*n*' into the memory at the address specified by '*VirtAddr*'. One of ordinary skill in the art would readily understand that under some conditions, the reading from and the writing to memory would occur from and to data buffer/cache 120 of Fig. 7.

27. The store value is described as being computed as:

$$(m_{127..64} \;\&\; m_{63..0}) \;|\; b \;\&\; \sim m_{63..0})$$

where subscripts are understood to mean extraction of a bit field (thus $m_{127..64}$ is the upper 64 bits of 'm', and $m_{63..0}$ is the lower 64 bits of 'm'), the '|' operator is understood to be a bit-wise logical-OR, the '&' operator is understood to be a bit-wise logical-AND, and the '~' is operator is understood to be a unary bit-wise logical-NOT. Since all elements of the store value computation are 64-bit values, each bit of the store value is computed according to:

$$H \ \& \ L \ | \ B \ \& \ \sim L$$

where 'H' is a respective bit of the upper 64 bits of 'm', 'L' is a respective bit of the lower 64 bits of 'm', and 'B' is a respective bit of 'b'; i.e. a two-to-one multiplexer selecting between 'H' and 'B' via control input 'L'. The store value computation is thus equivalent to a catenation of 64-bits of one-bit two-to-one multiplexers (hence the instruction is termed a Store Multiplex Immediate instruction), each multiplexer having a respective bit of the lower 64 bits of 'm' as a control input, and a respective bit of the upper 64 bits of 'm' as a first data input and a respective bit of 'b' as a second data input. The first data input (a respective bit of the upper portion of 'm') is selected when the control input is a logic one, and the second data input (a respective bit of 'b') is selected when the control input is a logic zero. One of ordinary skill in the art would readily understand that the computation of the store value would occur in ALU 102 of Fig. 7.

28. Thus the Store Multiplex Immediate instructions are described as forming a virtual address by adding contents of a register specified by a first operand to a sign-extended offset supplied as an immediate operand, reading a value from memory at the virtual address, multiplexing bit-for-bit the value with contents of an odd-numbered register specified by a second operand via a multiplexing control from contents of an even-numbered register specified by the second operand, and writing a result of the multiplexing to the memory at the virtual address.

29. At least Terpsichore manual pages 24, and 150-157 describe all elements of Claim 1 (as amended), as described in more detail in paragraphs 29-35 of this declaration.

30. The element decoding a single instruction for writing data to memory is described by the Terpsichore manual, as annotated and summarized by paragraphs 22-25 of this declaration. The Store Multiplex Immediate instructions are clearly each single instructions, as evidenced at least by the dedicated operation codes "S.MUX.64.B.A.I" and "S.MUX.64.L.A.I", and are clearly for writing data to memory, as they are store instructions.

31. The element based on a mask and data contained in at least one register is described by the Terpsichore manual, as annotated and summarized by paragraphs 26-27 of this declaration. The store value is dependent on the lower 64 bits of 'm' (from a register specified by an operand of the Store Multiplex Immediate instruction) that correspond to a mask, as each respective bit, when a logic one, masks off a corresponding bit of 'b'. The store value is further dependent on the 'b' value (from another register specified by the operand of the Store Multiplex Immediate instruction) that corresponds to data, as each respective bit replaces, when a corresponding control bit is a logic one, a corresponding bit read from memory.

32. The element the mask comprising fields that each correspond to a field of the data contained in the at least one register is described by the Terpsichore manual, as annotated and summarized by paragraphs 26-27 of this declaration. The bit-wise computation of the store value uses each of the lower 64 bits of '*m*' to select, bit-for-bit, between two 64-bit values (the upper 64 bits of 'm' and the 64 bits of the '*b*' value). Thus each bit of the lower 64 bits of '*m*' comprises a mask field and each bit of '*b*' is a corresponding data field.

33. The element detecting some of the fields of the mask as having a predetermined value is described by the Terpsichore manual, as annotated and summarized by paragraphs 26-27 of this declaration. The bit-wise computation of the store value effectively selects for writing corresponding bits based on respective control input bits that are a logic one (i.e. a predetermined value).

34. The element identifying corresponding fields of the data contained in the at least one register as write-enabled data fields is described by the Terpsichore manual, as annotated and summarized by paragraphs 26-27 of this declaration. Each of the corresponding bits selected for writing are write-enabled data fields, in contrast with the corresponding bits not selected for writing that remain unchanged, and thus are not write-enabled data fields.

35. The element writing the write-enabled data fields to a specified memory location is described by the Terpsichore manual, as annotated and summarized by paragraphs 26-27 of this declaration. The store value is written to the virtual address that is a specified memory location.

36. Thus every element of Claim 1 (as amended) is described at least by the Terpsichore manual on pages 24 and 150-157. In addition, every element of Claim 1 (as amended) is also described by the '599 patent, at least by the Zeus manual on pages 19-20 (describing general registers), pages 123-125 (pseudo-code for Store instructions, including Store Multiplex instructions), and pages 128-130 (pseudo-code for Store Immediate instructions, including Store Multiplex Immediate Instructions).

37. Based on the above, I believe that a person of ordinary skill in the art would readily conclude that the disclosures of the '599 patent and the '840 patent each describe "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements, as recited in claim 1 (as amended) of the 10/757,866 patent application, in sufficient detail to conclude that the inventors had possession of the claimed invention, as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent, and that the disclosures of the '599 patent and the '840 patent each provide sufficient detail to enable a person of ordinary skill in the art to make and use the claimed invention (as amended) of the 10/757,866 patent application without undue experimentation as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent.

Summary and Closing:

38. The '599 patent, including the Zeus manual, provides sufficient information in sufficient detail describing the claimed invention (as amended) of the 10/757,866 patent application, that one of ordinary skill in the art would reasonably conclude that the inventors had possession of the claimed invention at the time of filing the '599 patent. Further, the '599 patent, including the Zeus manual, provides sufficient information regarding the subject matter of the claimed invention (as amended) of the 10/757,866 patent application to enable one of ordinary skill in the pertinent art to make and use the claimed invention without undue experimentation. In addition, the '840 patent, including the Terpsichore manual, provides sufficient information in sufficient detail describing the claimed invention (as amended) of the 10/757,866 patent application, that one of ordinary skill in the art would reasonably conclude that the inventors had possession of the claimed invention at the time of filing the '840 patent. Further, the '840 patent, including the Terpsichore manual, provides sufficient information regarding the subject matter of the claimed invention (as amended) of the 10/757,866 patent application to enable one of ordinary skill in the pertinent art to make and use the claimed invention without undue experimentation. Therefore, I believe that each of the '599 patent, including the Zeus manual, and the '840 patent, including the Terpsichore manual, provide adequate written description and enablement as required by 35 USC § 112 for the "decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements of claim 1 (as amended) of the 10/757,866 patent application.

39. I have had no communication with any of the inventors of the 10/757,866 patent application (Craig Hansen and John Moussouris) relating to any material in this declaration.

40. I have been hired as a consultant in connection with procedures before the United States Patent and Trademark Office (USPTO) regarding patents and patent applications assigned to Microunity Systems Engineering, Inc., including the media processor patent application. I am being compensated for my services at the rate of $325/hour. Other than acting as a consultant in connection with procedures before the USPTO, I have no interest or connection with Microunity Systems Engineering, Inc.

41. During my evaluation of the media processor patent application, I have been impressed by the thoroughness and overall high-quality of the Zeus and Terpsichore manuals. The manuals provide clear and unambiguous descriptions of media processing systems and are thorough and well-written. The manuals provide comprehensive descriptions of instructions in complete architectural detail. The information in the manuals would have been readily understood and easily accessible to software engineers coding the media processing systems, and hardware engineers implementing microprocessors for use in the media processing systems, and that is exactly what architecture reference manuals should be. This is not surprising, since the '599 patent and the '840 patent each include an architecture manual that is intended to enable hardware engineers to do exactly that – design, build, and implement a media processor that would include circuitry for "decoding a single instruction for writing data to memory based on a

mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register", "detecting some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields", and "writing the write-enabled data fields to a specified memory location" elements as described in the Zeus and the Terpsichore architecture manuals.

42. I hereby declare that all statements made herein are of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing therefrom.

Date: 8/15/2007

Korbin Van Dyke: K. Van Dyke

Address: 3343 Little Valley Rd.
Sunol, CA 94586